

METAMODELS BASED ONTOLOGY DEVELOPMENT TO ACHIEVE STANDARD-COMPLIANT PROJECT MANAGEMENT

M.Sc. Stoyanova.T.

Dept. of Industrial Automation, University of Chemical Technology and Metallurgy, Sofia, Bulgaria,
tonya.stoyanova@gmail.com

Abstract: The present paper illustrates the current state of affairs for most organizations involved in the sphere of Information technologies that run or strive to run projects in accordance with applicable ISO standards and CMMI. This paper presents the target state of affairs and its design with used technologies. Reaching the goal state is summarized by serialization of metamodels in XML, mapping/parsing of metamodels (in XML) to OWL2 DL (RL profile) ontologies (written/exchangeable in RDF/XML), storages of these semantic metamodels and models (ontologies/semantic data) in advanced RDBMSs with support of ontologies, R2RML mapping of existing project-related relational data to semantic data and storage of semantic data in semantic stores hosted in existing RDBMSs or migration to such ones, extraction of non-relational project-related data and mapping of these logical models to a physical (relational) model and integration in RDBMSs to centralize all project-related data all governed by the defined vocabularies (ontologies of CMMI for Development and ISO 15504/ 12207). Achieving the goal state enables stakeholders to query and infer from project-related semantic data to manage, analyze and assess projects against underlying standards. Through commonly-accepted adapters, semantic data, metamodels and models are exposed to external applications.

Keywords: SOFTWARE, PROJECT MANAGEMENT, SPEM 2.0 MODELS, UML, ONTOLOGY, CMMI,

1. Introduction

Various data representation formats that are defined by Semantic Web Group of the W3C consortium are widely considered and accepted by the community as the next step in the data management. A significant challenge in the data management is sharing and analysing data stored in various and independent from one another applications. Advanced business applications for management of semantic data, represented in RDF/S [1, 2] and/or OWL 2 [3] ontologies rely on the latest-generation database systems (Oracle Spatial and Graph (Oracle 11g/12c) [4], IBM DB2/Graph Stores) for management of relational and non-relational data to solve highly complex problems of national security and intelligence, natural sciences and geolocation/ geopositioning.

This publication presents an analogous complex solution in information technologies, more precisely defines and implements a new model in the software project management in order to ensure the compliance of these projects with international standards and de-facto standards ISO 15504 [5], ISO 12207 [5] and CMMI (Capability Maturity Model Integration) for Development [6].

The analysis of the current state of affairs in companies that manage their projects in compliance with above standards reveals that: the information is not centralised stored; data related to projects, processes and standards are not centralised stored and managed; real on-the-job application of processes and standards is not transparent and is rather difficult to track and monitor.

This paper is focused on reaching the target state of a company that has already integrated and implemented the above mentioned standards at a "machine" level – this stage comes after the formal certification of the organisation.

Reaching the target state is summarised in the following points: XML serialisation of the developed metamodels of the standards; subsequent parsing/transformation of the generated XML metamodels into OWL2 DL (Ontology Web Language 2 Description Logic) (RL profile) [7] ontologies (written in RDF/XML [8]. RDF/XML syntax and semantics are widely accepted as the container for exchange of OWL2 DL ontologies); storage of semantic metamodels and models, i.e. the semantic data, in advanced semantic-relational database management systems that are capable of managing semantic data and of drawing inference from semantic data; R2RML (RDB (Relational Database) to RDF (Resource Description Framework) Mapping Language) [9] mapping of the existing relational models of project-related data and the semantic metamodels and models; transformation of relational data into their equivalent semantic data using R2RML and subsequent storage in the semantic data store [4]; extraction of non-relational project and process-related data (if such data exist in

the organisation and are distributed in various formats outside RDBMSs) and transformation of their logical models into relational models and integration of resulting models and data into RDBMSs (Oracle Spatial and Graph, IBM DB2 NoSQL Graph Store). Implementing the described steps will reach the goal of centralising project and process-related data and subjecting them to the unified ontologies describing CMMI for Development [6] and ISO 15504/ 12207 [5]). When the target state of the organisation is achieved, stakeholders will be able to query project and process-related semantic data and draw inference from semantic data in order to manage, analyse and assess projects against underlying standards. Moreover, centralised semantic data can be accessible to external (or third party) applications through adapters [10, 11].

2. Transformation of metamodels and project models into ontologies and semantic enrichment

2.1. Transformation of standards meta model and project models into ontologies

Another paper ("Development of Software Process Metamodels using SPEM 2.0"), by the same author, presented the skeleton of the SPEM 2.0 [12] metamodel of CMMI for Development ver. 1.3 [6], based on which a hybrid metamodel was developed that meets the requirements of both standards [6, 5] in order to achieve *Capability Level 3 Defined & Maturity Level 3 Defined* acc. to [6] and *Level 3 Established* acc. to [5]. The paper analysed the restrictions blocking the machine execution of metamodels and models developed with tools for business modelling that have a semi-formal architecture. Solutions for overcoming these restrictions were proposed and are focused on enriching the semantics of the metamodels and building their behavioural semantics. Solutions for validation of metamodels – in terms of completeness, accuracy and execution – were also proposed, i.e. validation using ontologies. The technical implementation (the integration of their ontologies) into the practice was also discussed. The present publication discusses the solutions at a much deeper level, introduces the technologies and the products that are used to build the solutions. SPEM 2.0 is based on the Model-Driven Architecture (MDA) standard that is used for defining the processes of the development of IT and software products and systems and their components. The construction of ontologies of SPEM 2.0 metamodels relies on the fact that both OWL [3, 13] and SPEM 2.0 are serialisable to XML. Constructing an OWL2 DL ontology of the defined SPEM metamodels and models would guarantee the completeness of metamodels and models and the time-bound computation of the inference from data and decisions. The OWL 2 RL profile is chosen with a view to the

current technical realisation [4] and the interoperability. The OWL2 RL profile is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate OWL 2 applications that can trade the full expressivity of the language for efficiency, as well as RDF(S) applications that need some added expressivity. The ontology consistency, class expression satisfiability, class expression subsumption, instance checking, and conjunctive query answering problems can be solved in time that is polynomial with respect to the size of the ontology. On the other hand, every OWL 2 ontology, which is an instance of the OWL2 DL structure, and, according to [14], can be translated (transformed) into RDF (Resource Description Framework) graphs (RDF graphs are the most widely used format for representation and storage of semantic data and most advanced RDBMSs use RDF graphs to store semantic data [4, 11, 15]) using bi-directional OWL 2 - RDF mapping [14]. Two formats for translation into OWL 2 DL ontologies are considered: OWL 2 XML [13] serialisation and RDF/XML, the latter is the generally accepted syntax for exchange of OWL 2 ontologies. The OWL XML serialisation is an alternative concrete syntax for OWL 2 ontologies and their exchange, while RDF/XML is the only normative concrete syntax for exchange of OWL 2 ontologies. The OWL XML serialisation accurately reflects the normative structural specification of OWL 2 and is defined in an XML schema that is used for the translation. The XML schema for the serialisation covers the structural specification of OWL 2 in its integrity, which also includes the OWL 2 language profiles. Alternatively, RDF/XML defines the XML syntax for representation of RDF graphs. The translation of OWL 2 ontologies into RDF graphs is done by constructing RDF-Triples. Every RDF graph consists of nodes and labeled directed arcs that connect pairs of nodes which are actually the set of RDF triples. Each RDF triple, at its turn, consists of a subject node, a predicate and an object node. Nodes in graphs are either IRIs (Internationalized Resource Identifiers), literals or blank nodes. Predicates are IRIs and are interpreted either as links between nodes or as an object node of a specified subject node. To encode RDF graphs to XML, nodes and predicates must be represented using the XML syntax and semantics, i.e. using element names, attribute names, element contents, and attribute values. RDF/XML substitutes IRIs for XML Qualified Names. The mapping of the serialised XML metamodels and models and the schema of the OWL XML serialisation and/or RDF/XML uses XSLT (XSL Transformations) [16] templates and applies them to source XML trees in order to generate resulting trees in the desired XML format. XSLT transformations define rules for generating the resulting tree from the source tree. Transformations are in fact associations of source code patterns and templates of transformations. Source code patterns are matched against elements of the source tree until templates are instantiated to generate the desired part of the resulting tree. Source and output trees are completely independent of one another. The structure of the resulting tree is independent of the one of the source tree; the resulting structure can be re-ordered, i.e. already transformed source elements can be re-ordered to fit in the desired structure of the resulting tree. Source XML trees of metamodels and models that were translated into RDF/XML, as it is the case, or to the XML schema of the serialisation of OWL 2 DL using XSLT are further loaded into the working environment [http://protegewiki.stanford.edu/wiki/WebProtegeAdminGuide, http://protegewiki.stanford.edu/wiki/WebProtegeUsersGuide] that supports RDF/S / OWL 2 DL ontologies. Resulting RDF/XML metamodels and models are translated into OWL 2 RL using the mapping in order to enrich the semantics of the ontologies (to elaborate the ontology at length using the expressive power of the semantics of [10, 12]). The OWL 2 RL ontologies are serialised back to RDF graphs (RDF/XML notation) using the mapping [14] to enable to be loaded into the semantic store [4].

2.2. Enriching the semantics of ontologies

Models must be complete and executable but first their semantics must be enriched to support the needed concepts and

meet the following constraints: each task/activity must start at a (pre) defined point in time; all tasks/activities must be time-bound and finish at a (pre) defined point in time; all suspended tasks/activities must be resumed and completed at (pre) defined point in time; task/activities cannot start and finish unless they meet defined constraints on the sequence of tasks and activities; each task/activity must be completed within the (pre) defined time interval; the process cannot be completed unless all its constituent tasks and activities are completed within the specified time interval allotted to each constituent. Semantics and constraints analogous to the described above must be defined for work products as well. Described states must be defined and present in the behavioural model of the metamodel. The definition of a task/activity must include its possible states, i.e. *not started*, *started*, *suspended* and *completed*. For example, these states can be modelled by introducing a new datatype and a data range constraint in the OWL2 ontology: (Declaration (Datatype (a:State)) DatatypeDefinition (a:State DatatypeRestriction (xsd:string xsd:enumeration "notStarted" ^^ "Started" ^^ "Suspended" ^^ "Completed"))) DataPropertyRange (a:hasState a:State)); a data property link is defined that connects individuals with the literal. As a consequence of having introduced states, the ontology must be enriched with temporal semantics to model the concept of time and timers associated with each task/activity. This is the prerequisite for defining rules of transitions from one state to another. The solution proposes the integration of the freely available OWL ontology that models the notion of *time* [17, 18]. In OWL-Time [18], the time ontology based upon the Web Ontology Language (OWL), *Instant* and *Interval* are basic mereological individuals, serving as foundational temporal entities. The OWL-Time ontology provides the vocabulary to encode topological relationships between time points and time intervals and expresses *duration* and *datetime*. However, the introduction of the concept *time* complicates RDF graphs, the search in semantic data and the inference.

The *time* ontology is expressed in OWL DL, which is a First Order Logic restriction, based on *SHOIN* DL (Description Logic) [19]. This description logic is decidable thanks to well defined semantics and proven reasoning algorithms which makes the *time* ontology an excellent candidate to use for the operations of the interval algebra [17, 20, 21]. The operations and the axioms of the interval algebra are used to express the *time* characteristics and their evolution. At the core of this algebra is the relationship between time intervals [17, 20]. Having two time intervals *TimeInt1* and *TimeInt2*, a time point *t* and a proposition, one might ask a variety of questions over the time domain and infer if the proposition holds [17, 20]:

- Mereological or “part-of” questions – for example: Is the time interval *TimeInt1* a sub-interval of *TimeInt2*? Does *t* occur within time interval *TimeInt1* or within *TimeInt2*? Is the time interval *TimeInt1* equal to the time interval *TimeInt2*?
- Topological or “connects” questions – for example: Does the time interval *TimeInt1* happen before or after the time interval *TimeInt2*? Do time intervals *TimeInt1* and *TimeInt2* meet?; Do these time intervals start and/or end at the same moment?;
- Logical or “rule-based” questions – for example: Does the proposition hold within *TimeInt1*?; If the proposition holds during *TimeInt1*, does it hold during *TimeInt2* too?.

These interval relations can be formally defined in terms of *before* relations among their beginning and end points [22].¹ Source metamodels and models and their ontologies respectively may now be considered complete compared with the initial state and discussed shortcomings.

[1, 26] A sample definition of a relation for overlapping time intervals:
 $\text{intOverlaps}(\text{TimeInt1}, \text{TimeInt2}) \equiv [\text{ProperInterval}(\text{TimeInt1}) \wedge \text{ProperInterval}(\text{TimeInt2})]$
 $\wedge (\exists t_2, t_3) [\text{ends}(t_2, \text{TimeInt1}) \wedge \text{begins}(t_3, \text{TimeInt2}) \wedge \text{before}(t_3, t_2)]$
 $\wedge (\forall t_1) [\text{begins}(t_1, \text{TimeInt1}) \implies \text{before}(t_1, t_3)]$
 $\wedge (\forall t_4) [\text{ends}(t_4, \text{TimeInt2}) \implies \text{before}(t_2, t_4)]]]$

3. Integration and implementation of semantic layer

The ontologies of metamodels and models that are integrated in the semantic-relational database system [4] are, in fact, the layer that envelops relational data and renders them highly normalised (compared with other approaches, commonly referred to as NoSQL approaches, that render relational data denormalised). After having integrated the ontologies, relational data and their semantic equivalents coexist, are extremely normalised and dependent on the ontologies.

3.1. Translation of relational data into semantic data

Existing relational data in RDBMSs are reused and the fundamental principles of Atomicity, Consistency, Isolation, and Durability (ACID) are preserved, which still ensure the security of data, the backup and recovery of data, the data compression, and the serial and parallel execution of processes. Existing process and project-related relational data are translated into their semantic equivalents acc. to the target vocabulary (the ontology) using R2RML (Relational to RDF Mapping Language). There is a straightforward approach to this mapping that does not use R2RML, however it is considered static and inflexible because of the structure of the resulting RDF graph; the structure of the latter strictly follows the structure of the database. The resulting RDF vocabulary copies the names of the elements from the database schema; the database schema and the output vocabulary cannot be changed. This approach being straightforward and static is inappropriate to be applied considering the complexity, the scale and the purposes of the described solution. Defining a mapping based on R2RML allows for the definition of highly customised views over relational data. Every R2RML mapping is tailored to a specific database schema and a target vocabulary. The R2RML mapping refers to logical tables to retrieve data from the input database. A logical table can be a base table, a view or a valid SQL statement, which is also referred to as an R2RML view since it emulates an SQL view without modifying the underlying database. Output RDF graphs that are generated based on the R2RML mapping use the predicates and the datatypes of the target vocabulary (the ontology). The semantic-relational data store [11, 4] can store a set of RDF graphs and record information for each graph to allow for the search for information spread in many graphs and the inference from many graphs as well. Logical tables are mapped to RDF using triple maps. Triple maps are the rules that map each row in a logical table to a number of RDF triples. The output semantic equivalents (RDF triples) of the existing relational data can be placed into named graphs (by default all output RDF triples belong to the default graph) using predefined graph maps which are also part of the triple maps (by definition, triple maps consist of optional graph maps, subject maps and predicate-object maps). The R2RML mapping itself consists of RDF in RDF/XML or Turtle (another concrete syntax for writing RDF/S ontologies). The transformation of the existing relational data into RDF graphs is carried out using the R2RML parser [23] (freely available: <https://github.com/nkons/r2rml-parser>), which supports a number of RDBMSs (Oracle, MySQL, PostgreSQL) as sources of relational data. The parser implements the R2RML standard and can generate output RDF graphs in Turtle, N-Triple [24], RDF/XML notations using the input R2RML mapping files. The coexistence of relational and semantic data in the same place brings the question of the consistency of the both types of data. Considering the huge amount of data, full re-parsing of all existing relational data upon constant changes in operational data is totally inappropriate. The consistency of both types of data can be ensured by incremental parsing of changed relational data based on transaction logs.

3.2. Loading semantic data into the semantic data store

The insertion of semantic data into the store may vary acc. to the format of semantic data [11, 4]:

- Bulk insertion of semantic data into the store using staging tables, where each row in the intermediary staging table contains components of an RDF triple - its subject, predicate and object, and optionally a named graph to which the RDF triple belongs (the SQL

loader utility is used to parse and load semantic data in N-Triple format into staging tables while semantic data in N-Quad [25] format are loaded into staging tables using external tables created with the SEM_APIS.CREATE_SOURCE_EXTERNAL_TABLE procedure; in both cases the SEM_APIS.BULK_LOAD_FROM_STAGING_TABLE procedure is called to populate staging tables with semantic data from external tables mapped to an N-Triple or N-Quad format input file);

- Batch insertion of semantic data using the RDF semantic graph support for Apache Jena; semantic data can be in RDF/XML, N-Triple and N-Quad format [<https://jena.apache.org/index.html>, 24, 25];
- Straight forward insertion of RDF-Triple semantic data into operational model tables using the SQL INSERT statement and the SDO_RDF_TRIPLE_S constructor that creates an RDF triple and assigns it to the model-graph to which it belongs.

3.3. Creating ontologies and loading them with semantic data

Semantic technology models (ontologies) [11, 4] are created with the SEM_APIS.CREATE_SEM_MODEL procedure and after that are loaded with semantic data using the INSERT statement. As far as the internal organisation and representation of models (ontologies) are concerned, the invocation of the SEM_APIS.CREATE_SEM_MODEL creates a model (ontology) identified by its unique name, a table (this table has already been created) that holds references to semantic technology data for this model, the name of the column of type SDO_RDF_TRIPLE_S in this table (this column holds references to semantic data created with the SDO_RDF_TRIPLE_S constructor), and a table namespace associated with the model (the dedicated table namespace has already been created). The creation and manipulation of models (ontologies) entail the creation and/or update of system views with metadata for models (the essential views are the MDSYS.SEM_MODEL\$ (a common view for all models) view and the MDSYS.SEMM_model-name (per model) view). An MDSYS.SEMM_model-name view with metadata (for the triples associated with the model) is created for each model (ontology). This view stores the following information for each triple: the identifier of the text value of the predicate of the triple, the identifier of the text value of the subject of the triple, the identifier of the text value of the canonical form of the object of the triple, the identifier of the text value of the object of the triple, the identifier of the model (ontology) to which the triple belongs, and the identifier of the text value of the graph name for the triple (*null* for the default graph).

3.4. Inference and rulebases

The selected semantic store [11, 4] can contain unlimited number of models (ontologies), rulebases and semantic data generated by the inference; the semantic store also allows the use of many rulebases, including the combination of the native inference and custom-defined rulebases, to draw inference from semantic data. The database [11, 4] inherently supports the following OWL vocabulary subsets (as they are described in the respective standards by the World Wide Web Consortium): RDFS, RDF (as a vocabulary subset of RDFS), OWLSIF, RDFS++, OWLPrime, and OWL2RL; each supported vocabulary has a corresponding rulebase, however these rulebases do not need to be populated because the underlying entailment rules of these vocabularies are inherently implemented. The native support of the OWL2RL language profile has motivated the choice of the semantic store and as discussed above this OWL2 vocabulary subset, which satisfies the requirements of the complex metamodels and models, is the selected syntax for the description of the metamodels and models. The domain-specific knowledge (i.e. the knowledge of how processes and projects are analysed and assessed acc. to standards [6, 5]) is transcribed into rules in custom-defined rulebases to provide the needed specialised inference capabilities. Custom-defined rulebases are created with the SEM_APIS.CREATE_RULEBASE procedure (for each rulebase, a system table is created to hold rules in the rulebase, along with a

system view `MDSYS.SEMR_rulebase-name` that is used to insert, delete and modify rules in the rulebase and (the view) has the following structure: the name of the rule, IF side pattern for the antecedents, filter condition that further restricts the subgraphs matched by the IF pattern, THEN side patterns for the consequents, and ALIASES that are one or more namespaces to be used, in addition to the default namespace, for expansion of qualified names in the query pattern). The `SEM_MATCH` table function is used to query semantic data; it requires the creation of entailment objects first (an entailment is an object containing precomputed triples that can be inferred from by applying a set of rulebases to a set of models; an entailment must be created for each rulebase-model combination in the query). The `SEM_MATCH` table function is able to query and draw inference from virtual models, which are virtually any combination of models, graphs and entailments. `SEM_MATCH` queries can also be written in the SPARQL [11, 16] SELECT format, which the semantic store inherently supports.

3.5. Validating ontologies, entailments and semantic data

OWL ontologies may contain errors, such as unsatisfiable classes, instances belonging to unsatisfiable classes, individuals asserted to be the same or different at the same time. Such inconsistencies in models (ontologies) and entailments can be detected and eliminated using the `SEM_APIS.VALIDATE_MODEL` and `SEM_APIS.VALIDATE_ENTAILMENT` validation procedures to ensure the trustworthiness of the models (ontologies), semantic data and inferred decisions.

3.6. Ontology-assisted search in relational data

The expressiveness of the search in relational data and the relevance of returned results are considerably increased with the modified ontology-assisted search which is not exclusively based on the lexical match of attribute names and data. The database [11, 4] implements ontology-assisted queries with the `SEM_RELATED` and `SEM_DISTANCE` operators. These semantic operators query relational data based on the semantic relationship between the data in a table column and terms in an ontology. The `SEM_RELATED` operator retrieves rows based on semantic relatedness, while the `SEM_DISTANCE` operator returns the distance measures for the semantic relatedness so that the output returned by the `SEM_RELATED` operator can be ordered by the relevance of data acc. to distance measures.

3.7. Access to the semantic store using adapters

RDF semantic graph (the database) [11, 24] provides support for Apache Jena and OpenRDF Sesame, which both extend the semantic data management capabilities of the database and provide for building open semantic applications, however Apache Jena is of more interest for the present solution. RDF semantic graph [11, 24] support for Apache Jena provides a Java-based interface to Oracle Spatial and Graph RDF Semantic Graph by implementing Jena Graph, Model, and DatasetGraph APIs, which are used for the management of named graph data (quads). RDF semantic graph support for Apache also provides a array of network analytical functions on top of semantic data ([15] in general functions for manipulating nodes and leaf-level nodes in graphs, functions for search for paths in graphs and their costs, in-depth multi-aspect analysis of semantic data). Apache Jena supports SPARQL [25] and applications using the APIs of Apache Jena to access the semantic store are integrated with it though the Joseki HTTP server (which also supports SPARQL queries). Apache Jena, with the capabilities and extensions it provides, is selected for a future development of a semantic application for project management as a wrapper of the core solution presented here in order to make the core solution available to a larger audience of business users and project managers.

4. Conclusions

The present publication presented a fully fledged solution and its architecture for the native (machine) integration and application of the standards being discussed throughout the publication [6, 5].

The core solution provides for the analysis and assessment of processes and projects based on inference from process and project-related semantic data and models (the ontologies of the standards). The architecture of the solution is open and implements standards and recommendations of the World Wide Web Consortium and other (referenced) open technologies and tools and eliminates the need of vague proprietary middleware solutions for the translation of metamodels and models. Oracle Database 11g/12c [4] with support for semantic technologies, which is still freely available for non-commercial and research purposes, was selected as the best available product which can suit the chosen technologies, the complexity and the scale of the solution. Two paths for future development of the present core solution are considered: development of a Java-based semantic application which will wrap and extend the core solution in order to make it available to a wider business and managerial audience and translation of production rules in the custom-defined rulebases, which store the domain-specific knowledge, into a concrete RIF (Rule Interchange Format) dialect [26, 27], which will make the solution entirely portable between organisations.

5. References

1. RDF 1.1 Concepts and Abstract Syntax W3C Recommendation 25 February 2014.
2. RDF Schema 1.1 W3C Recommendation 25 February 2014.
3. OWL 2 Web Ontology Language Structural Specification and Functional - Style Syntax (Second Edition) W3C Recommendation 11 December 2012.
4. Oracle Database 11g Release 2, Oracle Database 12c.
5. ISO/IEC 15504: 2004/2008, ISO/IEC 12207:2008
6. CMMI for Development, Version 1.3., CMMI-DEV, V1.3. CMMI Product Team.
7. OWL 2 Web Ontology Language Profiles (Second Edition) W3C Recommendation 11 December 2012.
8. RDF 1.1 XML Syntax W3C Recommendation 25 February 2014.
9. R2RML: RDB to RDF Mapping Language W3C Recommendation 27 September 2012.
10. Oracle® Spatial and Graph, RDF Semantic Graph Developer's Guide 12c release 12.1 E51611-06 November 2014.
11. OMG, SPEM, Software & Systems Process Engineering Meta-Model Specification, Version 2.0.
12. OWL 2 Web Ontology Language XML Serialization (Second Edition) W3C Recommendation 11 December 2012.
13. OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition) W3C Recommendation 11 December 2012.
14. Oracle® Spatial Topology and Network Data Models Developer's Guide 11g Release 2 (11.2) E11831-02 December 2009.
15. XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999.
16. Leonard Petnga, Mark Austin, Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems, Conference on Systems Engineering Research (CSER'13) Eds.: C.J.J. Paredis, C. Bishop, D. Bodner, Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013.
17. Time Ontology in OWL W3C Working Draft 27 September 2006.
18. Allen J.F. (1983), Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11):832-843, 1983.
19. Ian Horrocks, Oliver Kutz, and Ulrike Sattler, The Even More Irresistible SROIQ, School of Computer Science, The University of Manchester.
20. Moszkowski et al. (1984), Reasoning in interval Temporal Logic and Tempura. In Proc. AMC/NCF/ONR Workshop on Logics of Programs, volume 164 of LNCS, pages 371 – 383. Springer, 1984.
21. Hobbs et al. (2004), An ontology of time for the Semantic Web, ACM transactions on Asian Language processing (TALIP), 3(1):66-85.
22. Mitrou, Kouis, Konstantinou, Incremental Export of Relational Database Contents into RDF Graphs, In 4th Int. Conference on Web Intelligence, Mining and Semantics (WIMS'14), Thessaloniki, Greece, 2014.
23. RDF 1.1 N-Triples A line-based syntax for an RDF graph W3C Recommendation 25 February 2014.
24. RDF 1.1 N-Quads A line-based syntax for RDF datasets W3C Recommendation 25 February 2014.
25. SPARQL Query Language for RDF W3C Recommendation, 15.01.08.
26. RIF Production Rule Dialect (Second Edition) W3C Recommendation 5 February 2013.
27. RIF RDF and OWL Compatibility (Second Edition) W3C Recommendation 5 February 2013.